Fast Hierarchical Clustering using Reciprocal Nearest-Neighbor Chain Algorithm

Pratyaksha Wirapati

Bioinformatics Core Facility
http://bcf.isb-sib.ch

Swiss Institute of Bioinformatics
http://www.isb-sib.ch

SIB Days 2009, Fribourg

Summary

Agglomerative hierachical clustering is a versatile workhorse method for exploratory analysis of multivariate data such as gene expression microarrays.

One limitation of commonly used implementations, such as `hclust` and `agnes` in R, is that the algorithmic complexity is $O(n^3m)$, where n is the number of items to be clustered and m the number of features. Thus, it is not practical to cluster more than few thousands items, such as all the genes in a typical microarray dataset.

I implemented an alternative algorithm based on *reciprocal nearest-neighbor chains* which has algorithmic complexity of $O(n^2m)$ and produces the same tree as classical hierarchical clustering algorithm for linkage functions with *reducibility property*.

Typical microarray data matrix with hundreds of arrays and $\sim$20,000 probes can be clustered a few minutes on a typical desktop or laptop computer.

Hierarchical Clustering Algorithms

The classical "brute-force" algorithm for agglomerative hierarchical clustering starts by treating each item as a separate cluster, and then succesively merging the closest pairs. This requires n steps, and finding the closes pair at each step requires $O(n^2)$. Hence, the cubic time complexity.

Priority queues may be used to cache pair ranking and reduce the complexity to $O(n^2 \log n)$, but not many implementations uses this. Furthermore, it is still desirable to reduce the complexity further.

Since the late 1960's it's known that $O(n^2)$ algorithm is possible for single-linkage distance between clusters, which makes the problem equivalent to solving minimum spanning tree problems. However, single-linkage clustering often produces counter-intuitive results. A quadratic-time average-linkage algorithm, if possible, would be more useful.

Reciprocal Nearest-Neighbors

The conceptual breakthrough came in the late 1970's (Bruynooghe 1977, Benzécri 1982)[1]. The main idea is that to obtain the clusters that will appear on the final tree, it is not important to have the best pair at every step. As long as two clusters/items are *nearest neighbors* of each other (hence, "reciprocal"), their merging will appear in the final tree.

The tree is incrementally constructed in no particular order, yet the final outcome will be the same.

[1] M. Bruynooghe 1977 Méthodes nouvelles en classification automatique des données taxinomiques nombreuses. *Statistique et Analyse des Données* **3**:24-42

J. P. Benzécri 1982 Construction d'une classification ascendante hiérarchique par la recherce en chaîne des voisins réciproques. *Les Cahiers de l'Analyse des Données* **7**:209-218.

Nearest-Neighbor Chain Algoritm

1. Start with an arbitrary item i_1
2. Find the chain

   $$i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow \ldots \rightarrow i_k$$

   where i_{j+1} is the nearest neighbor of i_j. Stop when the nearest neighbor of i_k is a cluster/item already in the chain.
3. Remove the reciprocal nearest-neighbor pair from the chain (typically the last two of the chain) and merge the pair.
4. Continue extending the chain and merging the reciprocal pairs until the chain is used up.
5. If there are still remaining items or clusters, start a new chain.

Implementation: `nclust`

Written as a C library (ANSI 99), called `nclust`.

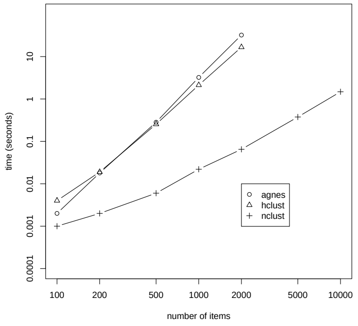
Uses several other optimization tricks: caching and queueing of distance between items/clusters, Lance-Williams update formula.

Distances between all pairs of items can be computed on-the-fly. Pre-computation of all pairs (huge memory for large number of items!) is not needed.

New data types (sparse, categorical, etc.) and the appropriate distance or similarity measures can be flexibly added as “plug-ins”.

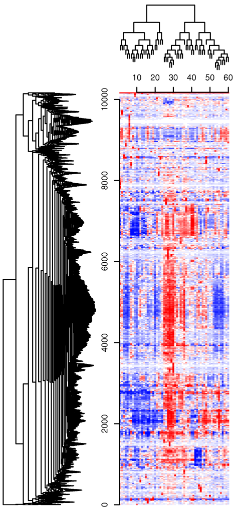
Binding as an R package, with fast implementation of large-scale dendrogram and heatmap plotting.

Benchmark



- Hardware: a laptop with Intel Core Duo (2.4GHz clock), only one core is used. 2 GB RAM.
- Toy problem: a simulated random matrix with varying number of items. Compare with `hclust` and `agnes` in R.
- `nclust` is not only faster, it's computation does not increase as quickly. Every 10-fold increase in the number of items requires 100-fold more time (unlike `hclust` and `agnes`, which require 1000-fold more time).

Real Data Examples

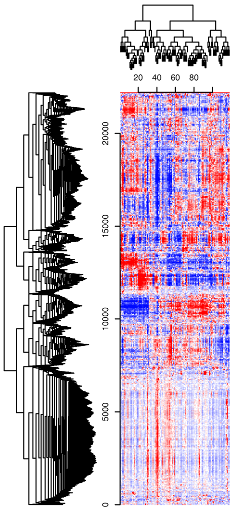


⇐

NCI60 drug response data: 60 cell lines (columns) and 10168 compounds (rows). Comput. time: 12.3 secs (On the same hardware as in previous plot)

⇒

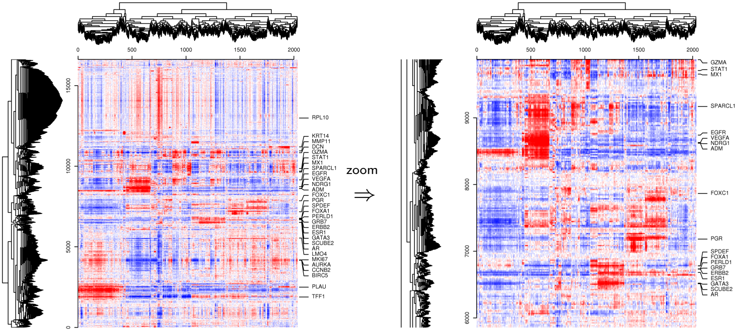
Breast cancer microarray data (Chin 2006): 118 arrays, 22215 probesets. Comput. time: 103 secs.



Large Expression Dataset: ExpO (GSE2109)

Data from Expression Oncology Project (`http://www.intgen.org`)

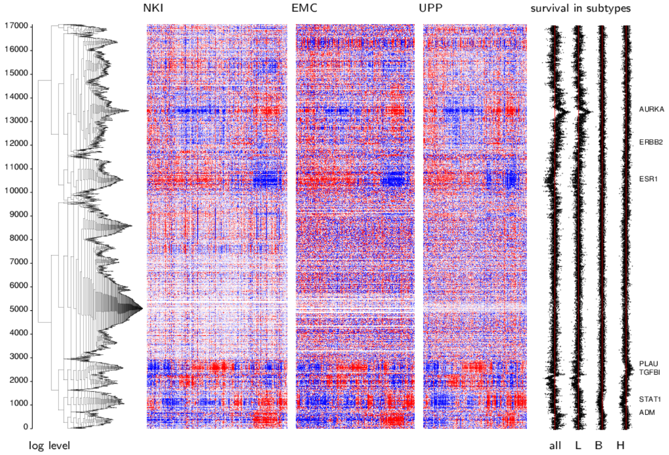
2035 tumors (various types), with 16634 non-redundant genes (after remapping to the NM subset of RefSeq).



Computation time: ∼11 minutes. Peak memory usage: ∼1 Gb

Breast Cancer Meta-analysis

Coordinated hierarchical clustering of 15 datasets (all co-analyzed; heatmaps of the other 12 datasets not shown)



⇐
Gene-by-gene Cox regression in all patients or subtypes

Subtype:
L: ER+/HER2-
B: ER-/HER2-
H: HER2+

2100 tumors, 17168 genes. Comput. time: ∼13 minutes

Conclusions

The practical number of items that can be handled by hierarchical cluster analysis is increased.

Ongoing works:

- Co-clustering across multiple independent and heterogeneous datasets
- Fast distance computation for sparse binary matrices, such as for GO and genesets/pathway data.